

REMARKS

In the Final Office Action¹, the Examiner provisionally rejected claims 8 and 18 under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 4, 12, and 19 of copending U.S. Patent Application No. 10/676,374 ("the '374 application"); rejected claims 1-4, 6-9, and 18-24 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent Application Pub. No. 2002/0108101 to Charisius et al. ("*Charisius*"); and rejected claims 10-12 and 14-17 under 35 U.S.C. § 102(b) as being anticipated by *Charisius*.

Claims 1-4, 6-12, and 14-24 remain pending in this application.

I. Provisional Double Patenting Rejection

Applicants respectfully traverse the non-statutory double patenting rejection of claims 8 and 18. Applicants request that the Examiner continue to hold the rejection in abeyance for at least the reason that no actual double-patenting circumstance can arise until a patent issues from the present application or the '374 application. Upon review of the remarks made in this paper, should the Examiner believe this application to be in condition for allowance but for the double patenting rejections held in abeyance, Applicants respectfully request that the Examiner contact the undersigned representative to discuss an appropriate resolution.

¹ The Final Office Action contains a number of statements reflecting characterizations of the related art and the claims. Regardless of whether any such statement is identified herein, Applicants decline to automatically subscribe to any statement or characterization in the Final Office Action.

II. Rejection under § 103(a)

Applicants respectfully traverse the rejection of claims 1-4, 6-9, and 18-24 under 35 U.S.C. § 103(a) as being unpatentable over *Charisius*. A *prima facie* case of obviousness has not been established.

“The key to supporting any rejection under 35 U.S.C. § 103 is the clear articulation of the reason(s) why the claimed invention would have been obvious . . . [R]ejections on obviousness cannot be sustained with mere conclusory statements.” See *M.P.E.P. § 2142, 8th Ed., Rev. 7 (July 2008)*. “The mere fact that references can be combined or modified does not render the resultant combination obvious unless the results would have been predictable to one of ordinary skill in the art” at the time the invention was made. *M.P.E.P. § 2143.01(III), internal citation omitted*. Moreover, “[i]n determining the differences between the prior art and the claims, the question under 35 U.S.C. § 103 is not whether the differences themselves would have been obvious, but whether the claimed invention as a whole would have been obvious.” *M.P.E.P. § 2141.02(I), internal citations omitted (emphasis in original)*.

“[T]he framework for objective analysis for determining obviousness under 35 U.S.C. § 103(a) is stated in *Graham v. John Deere Co.*, 383 U.S. 1, 148 U.S.P.Q 459 (1966) . . . The factual inquiries . . . [include determining the scope and content of the prior art and] . . . [a]scertaining the difference between the claimed invention and the prior art.” *M.P.E.P. § 2141(II)*. Office personnel must explain why the difference(s) between the prior art and the claimed invention would have been obvious to one of ordinary skill in the art.” *M.P.E.P. § 2141(III)*.

Independent claim 1 recites, in part, a computer program product operable to cause a processor to:

receive a first model in a first language from a storage device, the first model defining development objects representing building blocks for developing the application, relationships among the development objects, and constraints for developing the application;

generate a set of intermediate objects using the first model, wherein the set of intermediate objects comprises Java objects; and

generate an API using the set of intermediate objects as inputs such that the API enforces the relationships and the constraints defined in the first model and enables accessing the development objects.

The Final Office Action alleges that *Charisius* discloses, “receiv[ing] a first model in a first language from a storage device” (Final Office Action at page 5). The Final Office Action also alleges that “files defining classes and retrieved to build a OO model in terms of UML representations . . . reads on first model in first language” (Final Office Action at page 5).

Charisius discloses a “software development tool that creates a graphical representation of source code” (emphasis added) (paragraph 0057). *Charisius* further discloses, the “graphical representation of the project may be in Uniform Modeling Language . . . [A] developer . . . uses the software development tool to open a file which contains [the] . . . source code” (paragraphs 0088-0089). *Charisius* also discloses that the source code may be written in Java or C++ (paragraph 0089). Even assuming “graphical representation of the project . . . in Uniform Modeling Language (UML)” in *Charisius* could correspond to “first model in first language,” which Applicants do not

concede, retrieving a file which contains source code and then generating a graphical model (“in terms of UML representations”) does not teach or suggest “receiv[ing] a first model in a first language” (emphases added), as recited in claim 1.

The Final Office Action also alleges, “‘from a storage device’ is deemed fulfilled with Charisius retrieving of OO object definition files purport to implement a model” (Final Office Action at page 12). *Charisius* discloses that after opening a file which contains existing source code, “[t]he software development tool then obtains a template for the current programming language, i.e., a collection of generalized definitions for the particular language that can be used to build the data structureThe software development tool uses the template to parse the source code . . . , and create the data structure” (paragraph 0089). Retrieving a template (definitions) file for a source code file to parse the source code and create data structures as is done in *Charisius* does not teach or suggest “receiv[ing] a first model in a first language from a storage device,” as recited in claim 1. The Final Office Action also fails to actually indicate any storage device, as recited in claim 1, that is disclosed by *Charisius*.

The Final Office Action now seems to allege that paragraphs 0066 and 0093-0096, and Figures 12-19, of *Charisius* disclose “generat[ing] a set of intermediate objects using the first model, wherein the set of intermediate objects comprises Java objects” as recited in claim 1 (Final Office Action at page 6). This is also not correct.

Charisius discloses, “information can be extracted from and written to the models” (paragraph 0066). Paragraphs 0092-0096 and Figures 12-19 merely disclose various (graphical) views of an application, including the static view, the dynamic view,

the functional view, and the architectural view. As set forth above and in the previous response, *Charisius* discloses creating a graphical representation of source code. In contrast, claim 1 recites “generat[ing] a set of intermediate (Java) objects using a first model, wherein the set of intermediate objects comprises Java objects.”

The Final Office Action continues to allege that *Charisius* discloses, “generat[ing] an API using the set of intermediate objects as inputs,” as recited in claim 1 (Final Office Action at page 6). The Final Office Action now alleges, “definition source files along with graphical view of class symbols represented in UML model, code objects to generate an metamodel within interface 610, including instance of RWI, IDE or SCI **reads on** API being instantiated using intermediate objects” (emphasis in the original) (Final Office Action at page 6). This is also not correct.

As set forth above, paragraphs 0092-0096 of *Charisius* merely disclose various (graphical) views of an application. *Charisius* discloses, “static view is modeled using the use-case and class diagrams. A use case diagram 1200 . . . shows the relationship among actors and use cases within the system . . . class diagram 1300 . . . includes classes 1304, interfaces, packages and their relationships” (emphases added) (paragraph 0092). *Charisius* further discloses, “dynamic view is modeled using the sequence, collaboration and statechart diagrams . . . a sequence diagram 1400 represents an interaction, which is a set of messages 1402 exchanged among objects” (emphases added) (paragraph 0093). *Charisius* also discloses, a “statechart diagram 1600 includes the sequences of states 1602 that an object or interaction goes through during its life” (emphases added) (paragraph 0094). *Charisius* further discloses, a “functional view can be represented by activity diagrams . . . a special case of a state

diagram where most, if not all, of the states are action states” (emphases added)

(paragraph 0095). *Charisius* also states,

[an] architectural view . . . modeled using package, component and deployment diagrams. Package diagrams show packages of classes and the dependencies among them. Component diagrams . . . are graphical representations of a system or its component parts . . . Deployment diagrams 1900 show the configuration of run-time processing elements and the software components, processes and objects that live on them

(emphases added) (paragraph 0096). Displaying views of an application that show the relationships, interactions, and states of the objects of the application and how the application works as is done in *Charisius* does not teach or suggest “generat[ing] an API (Application Program Interface) using the set of intermediate objects as inputs”

(emphasis added), as recited in claim 1.

As set forth in the previous response, *Charisius* does not disclose or suggest using UML model to generate a metamodel within interface 610, including instance of RWI, IDE or SCI, as alleged by the Final Office Action. Also, as set forth in the previous response, *Charisius* does not disclose or suggest generating an API, as alleged by the Final Office Action. Rather, the “software development tool comprises” an API composed of an IDE, RWI, and SCI (paragraph 0064). *Charisius* discloses using the API and its components, IDE, RWI, and SCI, as part of the software development tool, **not generating** an API or any of its components. Accordingly, *Charisius* does not teach or suggest “generat[ing] an API using the set of intermediate objects,” as recited in claim 1.

In view of at least the above deficiencies of the *Charisius* reference, the Final Office Action has neither properly determined the scope and content of the prior art nor properly ascertained the differences between the prior art and the invention of claim 1. Accordingly, the Final Office Action has failed to clearly articulate a reason why claim 1 would have been obvious to one of ordinary skill in the art in view of the prior art. Therefore, a *prima facie* case of obviousness has not been established with respect to claim 1 and the rejection under 35 U.S.C. § 103(a) must be withdrawn.

Independent claim 18, though of different scope than claim 1, recites similar elements, and is thus allowable over *Charisius* for at least similar reasons as claim 1. Claims 2-4, 9, and 19-24 depend from independent claims 1 and 18, and are thus allowable over *Charisius* for at least the same reasons as the independent claims.

II. Rejection under § 102(b)

Applicants respectfully traverse the rejection of claims 10-12 and 14-17 under 35 U.S.C. § 102(b) as being anticipated by *Charisiu*. In order to properly establish that *Charisiu* anticipates Applicants' claimed invention under 35 U.S.C. § 102, each and every element of each of the claims in issue must be found, either expressly described or under principles of inherency, in that single reference. Furthermore, "[t]he identical invention must be shown in as complete detail as is contained in the . . . claim." See M.P.E.P. § 2131, quoting *Richardson v. Suzuki Motor Co.*, 868 F.2d 1126, 1236, 9 U.S.P.Q.2d 1913, 1920 (Fed. Cir. 1989).

Independent claim 10 recites, for example, "receiv[ing] a first model in a first language from a storage device, the first model defining development objects

representing building blocks for developing the application, relationships among the development objects, and constraints for developing the application, wherein the first language comprises unified modeling language.” *Charisius* does not disclose at least these elements of claim 1.

For reasons similar to those set forth above, *Charisius* fails to teach or suggest “receiv[ing] a first model in a first language from a storage device . . . wherein the first language comprises unified modeling language,” as recited in claim 10.

Accordingly, for at least the reasons stated above, *Charisius* cannot anticipate claim 10. Claims 11, 12, and 14-17 depend from independent claim 10, and are thus allowable over *Charisius* for at least the same reasons as the independent claim.


In view of the foregoing, Applicants respectfully request the Examiner’s reconsideration of the application, and the timely allowance of the pending claims.

Please grant any extensions of time required to enter this response and charge any additional required fees to Deposit Account 06-0916.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.

Dated: February 26, 2009

By: 
Eli Mazour
Reg. No. 59,318
direct telephone: (202) 408-4320